UCSD P-System

for the Osborne 1 Computer

by Thom Hogan

THInc.
Palo Alto, CA

Revision A

Table of Contents

A. Introduction

A.1 Welcome to UCSD P-System

Congratulations on your purchase of a double-density equipped Osborne 1 computer. With it, you have received the UCSD P-System, a special operating system that opens up a new world of software to your Osborne computer.

You newcomers will soon learn about all the features of your acquisition--we'll make the learning easy. Those of you who already have experience with the P-System will want to flip the pages and find the the section on making copies of your Pascal diskettes.

This manual will be your guide to learning about the P-System. This manual has been written with the novice user in mind; you'll find that we define new terminology as it is encountered and that we describe things in a straightforward, matter-of-fact manner.

Should you need further help in learning about the P-System, we suggest the following book:

> **Introduction to the UCSD p-System** by Charles Grant and Jon Butah, Bekeley, CA: Sybex, 1982.

We hope that you enjoy your introduction to the UCSD P-System; it is a powerful tool that can be used to make your microcomputer more useful.

A.2 Overview of this Manual

There's a lot to learn about the UCSD P-System, so you'll find a lot of information packed into this small manual. Fret not, however, as you'll find that the organization here is simple and to the point.

This first section describes what UCSD Pascal and the P-System are, and why you might be interested in it. We'll tell you what you need to use the P-System, how to make copies of your diskettes, and how to create a few extra, blank diskettes. In addition, you'll create a special diskette that contains your copy of the P-System. By the end of this chapter, you'll be ready to use software written on the P-System.

The second section of the book teaches you some details about the P-System, the operating system used by the UCSD Development System. We'll introduce you to the special language and notations involved with UCSD Pascal. You'll learn about files, some of the more common P-System utilities, and other useful tools that will help you use your new software.

The last section of this manual describes how to use the P-System with software you receive written in UCSD Pascal. You'll also find that the Universal diskette format and its use on the Osborne is described.

So, find a comfortable place to read, relax, and read on.

The remaining sections of the Introduction get you quickly prepared to begin using the P-System. We'll tell you what UCSD Pascal, the UCSD P-System, and the Development System are, what you should have to use your new acquisition, and how to make copies of the diskette you received.

A.3 Overview of UCSD Pascal and the P-System

UCSD Pascal has a short, but fascinating history. Developed at the University of California at San Diego beginning in 1977, the project was designed by Ken Bowles, then head of the computer science department there. A graduate student, Mark Overgaard, was the principal implementer.

The idea behind UCSD Pascal and the "P-System" is really quite simple. The problem with teaching computing, Bowles thought, was that the hardware (equipment) schools were using kept changing. Every time the hardware changed, someone had to redo the software that made the machines function. One result of this constant revising was that computer languages, the mainstay of computer science courses, kept changing in subtle ways as they were modified to work on new computers.

What Bowles envisioned was a language that remained the same no matter which machine it was used on. This is not the straightfoward task that it at first seems, however.

Computers differ in their internal structure (how they process and handle information) and by the type of central processing unit (CPU) they use. The CPU is the component through which all data must pass, sometimes to be modified or acted upon, sometimes simply to be moved elsewhere in the system. The instructions that are given to the CPU for the same function often differ from CPU to CPU. That means that if you write a program that has instructions for CPU "A", it is unlikely that it will work on CPU "B".

To get around this problem, Bowles came up with what he called a "pseudo-system," or P-System. In it, every instruction to the CPU is coded into a "pseudo-instruction." An "interpreter" program is used to look at each pseudo-instruction and convert it into instructions that the CPU can use. In other words, it's as if you programmed in a code ("P-code") and then used another program, the interpreter, to translate the instructions.

What the concept of the P-System did was to standardize the operating system and language Bowles used. As each new machine arrived, all that had to be reprogrammed was the interpreter, and this was not only a much smaller task than reprogramming everything, but was a task that could be reduced into a formalized process that could be handed to almost any competent programmer.

The P-System grew in popularity very quickly, especially in 1979 when Apple Computer decided to offer it as an option. The University of California began showing a healthly profit on the project; so much so, that the Internal Revenue Service stepped in. The IRS told the university Board of Regents that the large profits being shown by the UCSD Pascal project had perked their interest, and that they would soon ask that all California schools file separate tax returns, unless the university somehow split itself from the project or dropped it completely.

But UCSD Pascal and the P-System had already gotten so popular that there was no possibility that Bowles would let it die. Instead, the project was licensed to a separate company, Softech Microsystems. Bowles, because of conflict of interest rules established by the state, could no longer have any interest in this outside project, although Overgaard did join Softech as the head of program development.

Since the split, Softech has continued to improve the P-System. The version distributed today is referred to as version IV (i.e., four). This version is available for almost every CPU, whether it be a microcomputer, a minicomputer, or a large mainframe computer.

To understand what the P-System is, you must know more than its history; you must also know something about the various components it consists of.

The main core of the P-System is the operating system, the unit that allows all of the components of the computer to "talk" with one another. When you put your diskettes into the computer and start up the P-System, an interpreter appropriate for your CPU is loaded into the computer and executed. This, in turn, loads into the computer some of the rest of the instructions that normally are referred to as the Development System. The P-System you receive includes only a subset of the development system--specifically, the utilties needed to copy and execute program code from diskettes you receive from Osborne or other companies that have programs written in UCSD Pascal on them.

The Development System is the entire group of programs that allow you to develop programs. Osborne sells this in the Approved Software program. The Development System includes such "utility" programs as the Filer, which allows you to manipulate individual files of instructions or data (also included with your P-System diskette); the Editor, with which you create or edit files of instructions or data; the Compiler, which turns the instructions you type into executable programs; and a number of other smaller, miscellaneous modules.

When we refer to the Development System in this manual, we are referring the entire set of programs and instructions that make up the UCSD Pascal operating system. When we refer to the P-System, we refer to the subset of the Development System that you received with your double density option.

You might wonder where Pascal fits in?

One of the programs included in the optional Development System is the UCSD Pascal compiler. You use this compiler to convert your instructions into runnable computer code. The UCSD version of Pascal is similar to the International Standards Organization (ISO) definition of the language, but not entirely the same. The programs you run using the P-System were written in UCSD Pascal.

Pascal is not the only language that has been made available on the Development System. BASIC, FORTRAN, and COBOL are all also available, and the Development System normally includes an assembler for the CPU it is sold for. The pseudo-instructions for each of the languages is the same, the method of compilation from human-readable to computer-usable instructions is different for each of them, however.

By the time you've finished reading this manual, you'll ~~know~~ know all you need to know to run programs that use the P-System.

A.4 What You Need to Proceed

In addition to this manual you should have received at least the following items:

1 master P-System diskette
1 master Double Density diskette
1 double density manual

Of course you'll need your double-density equipped Osborne 1 computer, and we strongly suggest that you also have a printer available for use, since much of the software that runs with the P-System utilizes a printer. A parallel printer, such as an Epson, Okidata, or Centronics makes the most sense, but almost any serial or Centronics-compatible parallel will do.

We also suggest that you have at least four double density certified diskettes available before proceeding. You'll use three of these diskettes to create copies of the masters you were provided, while the other three will become your "work" diskettes. We'll describe how to make copies and format diskettes in the following section.

9 7 3+3

But enough introductory material, let's actually try out your new acquisition.

A.5 Making Copies of Your Diskettes

To make copies of your P-System diskettes we're going to have to start the system using one of the master diskettes you were provided.

Since there is a chance that you could accidentally damage a master diskette and not be able to proceed, you should follow our instructions carefully and completely. Don't allow yourself to get sidetracked and investigate other possibilities before you've copied your valuable master diskettes. Once copied, the master diskettes should be placed in a dry, cool, safe location, preferably one that provides protection from dust, water, and other everyday things around the house.

Find the diskette labeled **Double Density CP/M Upgrade.** & check label

Turn on your Osborne, or press RESET on the front panel, then carefully put your Double Density CP/M Upgrade diskette into the lefthand drive (if you're new to computing and are trying to use the P-System before learning how to use the computer, you should first work through the first three chapters in the Osborne 1 User's Guide). Close the drive door after you've inserted the diskette.

Press the key labeled RETURN on the Osborne 1 keyboard. The disk activity light comes on, a few moments pass, and then you see the following message:

```
[ HELP program ]
```

Press the ESC (for escape) key, then type COPY followed by another RETURN. Follow the directions in the Double Density Users' Guide to copy diskettes. Copy both the Double Density CP/M Upgrade and the UCSD P-System diskettes. Here's an abbreviated list of the steps you'll go through if you're doing everything correctly:

1. Press RESET or turn on computer.
2. Put Double Density CP/M Upgrade diskette in drive A.
3. Press RETURN.
4. Press ESC.
5. Type COPY and press RETURN.
6. Press C, to copy.
7. Insert blank diskette in drive B and diskette to be copied in drive A.
8. Press A to indicate the original diskette is in drive A.
9. Press RETURN to start the copying.
10. Repeat steps 7 through 9 for each diskette you wish to copy.

While you're using the COPY program, you should make two blank, formatted double density diskettes. You do this by pressing RETURN to get back to the main COPY menu. Then, the following steps are performed:

1. Press F, to format.
2. Insert a new diskette in drive B.
3. Press B to indicate the diskette to be formatted is in drive B.
4. Press RETURN to start formatting process.
5. Press D to indicate double density.
6. Repeat steps 3 through 5 for the second diskette.

Make sure you label each diskette you copied or formatted so that you can identify each during the steps that are described in the next section.

If you have any questions about how to copy or format diskettes, you should refer to both your Osborne 1 User's Guide and your Double Density User's Guide for more detailed explanations.

Creating a Working P-System Diskette

You're now ready to begin your first venture into the use of the P-System. Let's make sure we all start from the same common point; do the following:

A. Press RESET on the front of the computer.
B. Take out all diskettes from the drive.
C. Put the copy you made of the UCSD P-System diskette into the lefthand drive.
D. Press RETURN.

should have new address

In a few moments you'll see the Osborne logo appear, along with the message "UCSD P-System." After a few more seconds, the following appears:

[PRIME screen]

"first" is spelled as "fist"
"blank" should be "blank, formatted" in both references

If you've followed the instructions in this manual up to this point, you are ready to proceed, so press Y and RETURN to verify that you are ready. If for some reason you've managed to get to this point of the manual without first making the copies we suggested, STOP! Press N and RETURN, then go back to page x and follow the instructions!

Put one of the diskettes you formatted into the righthand drive--we're going to make that diskette into your "working" P-System diskette (i.e., the one you use to run UCSD Pascal programs).

The menu gives confusing message: i.e. either "RETURN" or "Y"

S in stand by should not be capitaliz

Press RETURN to start the copying process. This will take several minutes, so relax or find something else to do while the computer is churning away creating your new diskette.

For the curious, what is happening is that a program named PRIME is being run. What PRIME does is to take 2Ø tracks of information from the UCSD P-System diskette you copied and place them in the proper place on the blank, formatted diskette. The P-System diskette we supply is really a hybrid diskette--half of it is in the CP/M format you're so used to, while the other half is in the P-System format. What the computer is now doing is making a diskette that is just P-System-oriented.

When the copying process is completed, a message to that effect appears, then CP/M is restarted. Just for the fun of it, type DIR and press RETURN to see the files that are on the UCSD P-System diskette we provided. You'll find the following:

[DIR OF UCSD diskette]

The file PBOOT.COM is used to start your Osborne 1 computer in the P-System format. We'll do so in the next chapter, but not until you've first learned a few things about the P-System itself.

B. The P-System Commands

B.1 Getting Around

In this chapter we're going to tell you about the terminology used in the P-System. We'll define words, introduce you to the P-System file naming conventions, and show you what you can do with the P-System as we provide it.

Prompts Defined

If you're used to another computer operating system--like CP/M--you are going to find the UCSD P-System method of presenting information to you quite different.

Most computer systems show you a "system prompt" and then wait for you to type a complete command. Here are a few examples:

Operating System	Prompt	Typical Command
CP/M, IBMDOS	A>	DIR [RETURN]
Applesoft BASIC	]	CATALOG [RETURN]
UNIX	$	LS [RETURN]

With the P-System, the concept of prompting the user for a command changes. Instead of the primitive prompts shown above, you'll see a more complex prompt, normally called the promptline or menu:

```
Command: E(dit, R(un, F(ile, C(omp,  ? [IV.Ø 3Bo]
```

The above promptline is offering you a finite choice of things to do. The P-System expects you to press a letter to indicate which of the possibilities you wish to perform.

For instance, if the main promptline shown above appears on your screen, you could press the letter F to tell the system that you wish to use the "filer." You need not press the RETURN key to verify your choice--the minute the system sees you press the F key, your request is accepted and acted upon.

The UCSD promptline concept is really rather simple. Each promptline consists of three basic components:

1. Description
2. Options
3. Version/Other Options

Let's look a typical promptline again, and separate out the three basic components:

Filer: G(et, S(ave, W(hat, N(ew, L(dir, ?[A]

The description on a promptline comes first. In our example the description is **Filer:**. In most cases the description ends in a colon, but some non-UCSD utilities may not use this convention.

What the description does is tell you which major module that you are using. If you are using the filer, the promptline reads **Filer:**; if you are using the editor, the promptline reads **Edit:** (note: the editor is not included with the P-System). If you are at the highest command level (as when starting), you'll see **Command:**.

After the description comes a list of options available to you. The capital letter that precedes each "(" is the letter to type to execute that option. The full option name--at least the full option name as it appears on the screen--consists of a capital letter, a left parenthesis, and the remainder of the option name. Here is the list of options that appear in the promptline we're studying:

Option on Screen	Press This Key	What it does
G(et	G	"gets" a file for use
S(ave	S	saves the current workfile
W(hat	W	identifies the current workfile
N(ew	N	starts a new workfile
L(dir	L	lists a directory of files

As you can see, the option names--with possibly the exception of **L(dir**--suggest the action that is about to take place (actually, even **L(dir** makes sense if you simply remember it as short for "list directory").

We'll find the use of a capital letter followed by a left parenthesis a common practice throughout the P-System, not just on the promptline, so make sure you understand the concept of separating the key to press from the rest of the name of the command (with the parenthesis) before moving on.

The last part of the promptline consists of a question mark (sometimes) and an odd representation enclosed within square brackets at the end of the line.

The question mark is simple to explain: if it appears, there are more prompts possible than can fit on the current line. Press the ? key to display the other commands that are available. In the case of our Filer example used above, another part of its promptline looks like this:

Filer: B(ad-blks, E(xt-dir, K(rnch, ?

Something to Remember: Just because an option prompt doesn't show on the top line of the screen doesn't mean that the option you specify won't be recognized. For instance, if the first Filer prompt we showed you appeared on your screen, you could press B, E, K, M, P, V, X, or Z and the action for these options would start. Pressing ? is simply a way of showing you the "other" options that are available at a given system level. If no question mark appears, than no other options are possible.

The numbers and letters that appear within the square brackets refer to the version of the current program you are using. The [IV.Ø 3Bo] that appears on the first promptline that appears when you use the P-System (the Command promptline), indicates that you are using version IV.Ø of the P-System. It is strongly suggested that you develop the habit of indentifying your programs in the same manner, as this is a simple and effective method for showing the "vintage" of a program. If you later receive an update to your software and the promptline shows [IV.1], this is your verification that you are using the correct version of the operating system, and haven't accidentally picked up an older version.

By the way, the promptline is not the only method with which options are presented to you. Sometimes options are presented in other manners--the most frequently used method is to present a "menu" of choices on the screen for you to choose from.

The top line prompts in the P-System are found in the highest level programs available to you, the master command menu, the filer menu, the monitor menu, and so on. When specific information is requested of you--as in the request for you to name the file you wish to work with--it is normally done just below the top promptline.

We could make our explanation of getting around in the P-System much more elaborate, but unless you are using the Development System, it isn't necessary to go into greater detail than is presented here and in the next sections. Suffice it to say that you could discover parts of the P-System that we don't discuss in this manual. If your curiousity gets the better of you, we suggest that you go ahead and purchase the complete Development System, or consult the book recommended in the Introduction, Introduction to the UCSD p-System.

B.2 Files

This section describes files--what they are, how they're named, and the concepts of the "workfile" and "volumes."

File Names

Within the P-System you'll be using diskette files. In brief, a file is an arbitrary method of storing related information, whether it be text, programming instructions, or information that the system uses to keep track of what is happening.

sp?

You are probably familiar with the concept of filing information in a box or file cabinet. Storing information you wish to later retreive with the computer is the same idea, only instead of having a physical container to hold the information (e.g., sheets of paper in a file folder), data the computer stores is kept in magnetic pulses on floppy diskettes, grouped as diskette files. We'll refer to these diskette files simply as "files" in the remainder of this manual.

Just as you must identify file folders in a cabinet in order to order the information and make it easier to retrieve, so must you name diskette files. We refer to such identification as the "file name."

In the P-System, file names can be of your own choosing, and consist of up to 15 characters (including the file type; see below). Any lowercase character you type for a file name is automatically converted to uppercase by the computer. If you include a blank in a file name, the P-System will not recognize it; the same holds true for control characters.

In addition to the standard alphabet, file names may also contain several special characters. The full list of characters that may appear in a file name are as follows:

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z / - _ .

File names can never use the following characters:

$: = ? ,

If you attempt to name a file using one of these special characters, the P-System may seem to recognize it, but the file name that is used by the system most likely will not be the same as the one you intended. The reason for this will become clear as you learn about some of the other aspects of the P-System. For now, just realize that typing $, :, =, ?, or , causes the system to assume something about your file name that you might not have intended.

One final comment about file names is in order. A tendency of many computer users is to use abbreviations or shorthand notation in naming files. You might see a file named **AP12**, for instance. Could you guess at what the contents of this file are, just from its name? We think not.

The tendency towards using short, cryptic file names comes from the "olden" days of computing, when the computer would only allows five or six letters to be used in naming a file. Obviously, with only five or six letters, the possibilities for giving a meaningful name to a file are extremely limited.

There is no reason that you should develop the bad habit of arbitrarily shortening file names. Instead, make sure that when you name a file, its name clearly reflects what is in the file. Our example of **AP12** could just as easily have been named **PAYABLES.12** using the P-System. The first portion of the file name--PAYABLES--clearly tells you what the file contains (unpaid bills for you non-accountants), while the number 12 is some indication that this is the twelfth file with payables information. The period between the two parts helps the reader identify the two separate pieces of information. (By the way, if the number 12 was meant to indicate the second day of the first month, then the file was still poorly named--it should have been named **PAYABLES.JAN.2** to correctly indicate its contents.)

To summarize, file names should clearly identify the contents of a diskette file, and can be up to 15 characters in length.

File Types

A second method of identifying files also exists. Since files can perform so many different functions within the P-System, we need some method of telling the computer what the contents of a file are to be used for. To do so, we use something called the "file type."

File "types" make it easier for the system to assume information about your files and what they are to be used for. For instance, if all programs always had the same file type, the computer might not know when to interpret a file as instructions and when to interpret it as data. The P-System gets around this problem by having several specific, predefined file types.

For the most part, you'll be working with the following file types:

File Type	What is in it
.TEXT	Normal, readable text material. Generally this information is formatted to be used by the Development System editor.
.BACK	A previous version copy of a .TEXT-type file. If you keep multiple versions of the same readable text file, the latest version should be in the file of type .TEXT, while the previous version of it should be in the file of type .BACK.
.CODE	Executable instructions, or "code," for the computer to use. The .CODE file type is used for both P-code and machine code in the Development System. The programs you receive from Osborne and other vendors normally come to you as .CODE-type files ready to run.
.DATA	Information that is stored for later use, not necessarily in readable form. Usually the file type of .DATA is reserved for information that is used by, or created by, a program you use on the computer.
.BAD	As you can probably guess, the file type of .BAD indicates that the information in it is not usable, or of no interest. In the P-System, the file type of .BAD specifically refers to any file that occupies an area of the diskette that the computer cannot access correctly (as in the case of a damaged diskette).

Other file types exist in the P-System, but they are primarily of interest only to users of the Development System, so we won't attempt to deal with them here.

You need not always identify the type of file you are working with. For instance, the Development System editor assumes that you would want to work with a .TEXT-type file, while the runtime interpreter included with the P-System assumes that the program name you type has the file type of .CODE.

When you do need to identify the file type, you do so by typing the file name, a period, then the file type. If you remember from the previous discussion of file names, the period is a valid character within a file name; how does the system know whether the information that follows a period is the file type or part of the file name?

The answer is simple: the last period detected in the file identification you present the computer is assumed to preceed the file type. This means that you should be careful about naming files with periods in them.

MY.TEXT	Is file MY of type .TEXT
MYTEXT.TEXT	Is file MYTEXT of type .TEXT
MY.TEXT.TEXT	Is file MY.TEXT of type .TEXT

The Workfile

A special file exists in the Development System, and is often found when using P-System software. If you are editing work without having associated a file name with it, it will be stored in a file named **SYSTEM.WRK** with an appropriate file type (normally either .TEXT or .CODE).

The purpose of the workfile is to allow you to actually edit and run programs without having to first commit to putting the information you're working with into a specific file. An example of when this might be desirable would be as follows: you are using the Development System and you have a file that already exists with old information in it that is usable, but out-of-date. You want the new version to have the same file name as the current one (perhaps others use the machine and you don't want to reteach them how to use the file). You can try out the new file by creating it as a workfile. Once you get it the way you want it, you rename your old file to have the file type of .BACK and rename the workfile to have the file name you desire.

Since the concept of the workfile is unique to the P-System, let us recap its use.

Normally, instead of giving a file a name when you first create it, you use the workfile (SYSTEM.WRK.TEXT) instead. Later, when your work is (almost) complete, you would give the file a valid name and type. In brief, the concept of the workfile is a way of working without having to commit to file names.

On a more subtle level, the workfile idea tends to make programmers think of program development in two stages: first the workfile stage, where early thoughts and work are not necessarily committed to; and last, the formalization of the program (or text file) and its structure by giving it a name and beginning to use it.

Volumes

So far we've only been discussing files--we've given you no idea of where they come from or where they are stored.

The P-System term "volumes" opens up the issue of input and output; where do files come from and where do they go?

A volume is any input or output device. The terms "input" and "output" both refer to the idea that information in a computer system moves from one portion of it to another. The screen is an output device, as is a printer. A keyboard is an input device. Modems and diskettes can be both input and output devices. In other words, information from a keyboard goes IN to the central computer circuitry, thus the term input. Information that has been processed by the computer goes OUT to printer or screen, thus they are output devices.

The P-System predefines a number of input and output devices. Here's a list of these:

Volume Number	Volume Name	Description of the Device
#1:	CONSOLE:	The screen and keyboard, typed characters are displayed.
#2:	SYSTERM:	The screen and keyboard, typed characters are not displayed.
#3:	GRAPHIC:	The screen when used for graphics (not implemented on the Osborne)
#4:	diskname	The diskette in the main disk drive, referred to as the "system disk."
#5:	diskname	The diskette in the second disk drive, referred to as the "alternate disk."
#6:	PRINTER:	The printer, if available.
#7:	REMIN:	A serial input line, as from a modem.
#8:	REMOUT:	A serial output line, as to a modem.

Devices #9: through #12: may also be available if additional disk drives are connected to the computer.

In the above chart, you should notice several conventions that are used in naming the devices. First, you may refer to any device by its number. You do so by typing

- o a number sign
- o the device number
- o a colon

You can also refer to devices by their name. You do so by typing

- o the name
- o a colon

Another thing might have caught your attention about the chart: the two disk drives names are not predefined. Every P-System diskette is given a name, which is stored on that diskette. It is this name that is used for the disk devices, not the word "diskname." For example, the Osborne version of the P-System is supplied with the following diskette:

OSBORNE:	The diskette for the main drive, sometimes called the "system diskette" because it is used to start the system.

We haven't yet told you for what purpose you use the volume identification. Actually, it's quite simple: the volume identifier tells the system where it is going to get a file from or where it is going to put it. If you wanted to use a file named **MYFILE.TEXT** that exists on the first system diskette, you would identify the file as:

SYS1:MYFILE.TEXT

r

You should now see the reason why a colon can't appear in a file name or file type: the computer would interpret the colon as being the market between the volume to use and the file name to find.

A shorthand method of specifying the system disk device (device #4:) is available--use an asterisk instead of the name. Thus, ***:MYFILE.TEXT** and **OSBORNE:MYFILE.TEXT** refer to the same file.

If you do not specify a volume name or number, a special "prefix" number is used. Normally, this prefix number is that of the system diskette (i.e., #4:), but it can be changed using the system filer's P(refix command. It is useful to set the prefix to the volume number of the most common device you are using, then you no longer have to always specify a device name in addition to a file name.

Putting it All Together

We've already presented the idea of volume, file name, and file type representing the entire specification of the information you want to use while in the Development System. But to make sure that you understand it completely, let's recap:

- o The complete specification for a file is the volume it is to be sent to or from, the file name, and the file type.
- o Under some circumstances, you may omit pieces of information. If you omit the volume, a default device is used. If you omit the file type, a

default file type is assumed. If you omit the entire file name specification, a workfile is used.

Wildcard Conventions

Users who are familiar with CP/M may be wondering if there is a shorthand method of referring to groups of files, as there is in CP/M. Yes, there is.

To review, CP/M uses the following "wildcard" conventions (so named because of their similarity to the Joker in poker games--a wild card that can be made into just about anything):

?	replaces a single character and tells CP/M that you do not care what letter is in that position in the file name.
*	replaces a group of characters and tells CP/M that you do not care what letter or letters are in that position in the file name. In fact, the asterisk (*) does not care how many characters it replaces--it could be any number of characters between one and eight.

In the P-System, two wildcard specifiers are also possible, although they have slightly different meanings then their CP/M counterparts. The wildcard characters are:

=	replace a group of characters without querying the user.
?	replace a group of characters and query the user.

In a sense, both of these P-System wildcards function similarly to the CP/M asterisk--they replace a group of characters. Just like in CP/M, only one "group" wildcard can appear for the file name, and only one can appear in the file type, although they can appear in any character position. The following are all valid uses of wildcards:

A=Z	use all file names that start with the letter A and end with the letter Z.
=	use all file names.
=RIGHTS	use all file names that end in "RIGHTS."
UANDIARE=.TO	use all file names that start with "UANDIARE" and have a file type of "TO."

Now for the querying part: If you use the question mark instead of the equal sign for the wildcard character, the action to be performed with that file will not be done until you have verified it by pressing the Y key. A message appears to indicate that you must verify the action. The action isn't performed if you press the N key.

Here's an example. You are removing files from a diskette using the filer. When the filer asks you which files to remove, you specify:

```
Remove what files? ?.?
```

followed by a RETURN. The filer next presents the following message (assuming the first file it found was "MISTAKE.CODE"):

```
Remove MISTAKE.CODE? (Y/N)
```

Press Y to remove the file, N to leave it on the diskette. This process continues until all files matching your wildcard specification have been exhausted.

B.3 The Modules Available

This section of the manual tells you about what portions of the UCSD Development System are provided to you with your P-System diskette and how to use them.

Filer

The filer is the function module that allows you to manipulate (do things) with files. You can rename files, transfer files from diskette to diskette, erase files, look at directories of files, and perform a number of more specialized functions.

Once you've pressed F, for filer, at the command level, the filer promptline appears. Your new choices are as follows:

B(ad blocks
C(hange
D(ate
E(xtended directory
G(et
K(runch
L(ist directory
M(ake
N(ew
P(refix
Q(uit
R(emove
S(ave
T(ransfer
V(olumes
W(hat
X(amine
Z(ero

Since the filer is an integral part of the P-System, we'll look at it a little closer than the rest of the system. You'll be using the filer frequently, so take the time to learn its commands.

Bad Blocks

The Bad Blocks function allows you to check the diskette for sections (blocks in P-System nomenclature) that are faulty. By faulty, we mean that the CRC number--a number that is calculated based upon the information that is stored in the block--doesn't match the new CRC calculated when the Bad Blocks option is in action. (CRC, by the way, stands for Cyclic Redundancy Check, a fancy way of saying self-calculating.)

Using the Bad Blocks option results in the following prompt:

    Bad block scan of what vol?

You reply with a valid P-System volume name. Next, the following displays:

    Scan for 270 blocks? (Y/N)

This means that there are 270 blocks on the diskette to be scanned, and the P-System wants to verify that you want all of them scanned. If you reply with a Y, scanning begins; if you reply with a N, you'll be asked:

    Scan for how many blocks?

You should reply with the number of blocks you want scanned.

Of what use is the Bad Block report? You can use it to attempt to resurrect the information in a block by invoking the Examine function, or you can isolate it from your good data by assigning any free space to a bad block by invoking the Krunch function.

Change

You may rename files using the filer's Change function.

Press C, for Change, while the filer promptline is displayed. You'll be asked

    Change what file?

to which you should reply with a file name (and volume name and type, if desired). You may use the wildcard functions, although we suggest that you use the query wildcard (?). Next, the system displays

    Change to what?

Again, type a valid P-System file name (or volume and type, or wildcard).

If you wish, you can change the name of a volume by only typing its name (i.e., not typing any file name after it).

A standard convention in the P-System allows you to skip the second message display by anticipating the P-System's request for more information. When the first prompt appears, type in your response, a comma, then the next response the P-System will want. If you wanted to change the file "OLD" into the file "NEW," you would type the following:

    OLD,NEW

followed by a RETURN.

Date

The P-System keeps track of the date when you save information into files. Obviously, then, you need some way of telling the system the date.

Press the ? key until you see the remainder of the menu prompt that includes **D(ate**. Press the letter **D**. Now you should see the following:

```
Date set: <1..31>-<JAN..DEC>-<ØØ-99>

Today is 21-Nov-82

New date?
```

The computer is waiting for you to enter a new date in the format indicated by the top prompt line (i.e., the day, a hyphen, a three-letter abbreviation for the month, a hyphen, and the last two digits of the year. If today is the third day of January, 1983, you would type:

3-JAN-83

followed by pressing the RETURN key to indicate to the computer that you are finished.

Once you've finished typing the date and pressed the RETURN key, the computer will indicate the date it has set on the line underneath the one you typed in the date on. If you make a typing or formatting mistake, note that the P-System uses the original date it had, not your new, incorrect one. The Date function <u>only</u> accepts information in the format specified above, any other results in no new date being set.

To leave the Filer once you've set the date, press the Q key. You should see the "Command:" promptline reappear.

We recommend that the first thing you do during any session using the P-System is set the date. If you do not do this, you lose a potentially useful piece of information about your files.

Extended Directory

An extended directory of files can be listed using the filer. You do so by pressing E while in the filer. The system will ask you which volume you wish a listing of files for by prompting:

```
Dir listing of what vol?
```

You may reply with any valid P-System volume name, any valid P-System volume number, or an asterisk to specify the current volume.

In addition, you may specify a file name to look for, or a group of files to look for (using the wildcard specifier). An example of this might be:

```
Dir listing of what vol? #4:MYFILES.=
```

If you want to save listing in a new file, you specify that file following a comma after typing the name of the volume or files you're looking for:

```
Dir listing of what vol? #4:MYFILES.=,LISTOFMYFILES
```

To send the list of files to your printer, type:

```
Dir listing of what vol? #4:MYFILES.=,PRINTER:
```

A related command is the List Directory command, explained later in this section.

Get

The Get function tells the system what file to use as the workfile (in place of the default workfile). This command is frequently used with the Editor and Compiler, two modules that are only present on the Development System diskette, and are not included with the P-System.

Krunch

The Krunch command consolidates the areas on a diskette to make the best use of a diskette.

After you've been using a diskette for awhile, the changes you've made to the files on that diskette may result in some wasted space. For instance, if you've erased a file, the space formerly used by the file is wasted. To remedy this situation, you use the Krunch function.

When you press K, for Krunch, the system responds with:

```
Crunch what vol?
```

You type the volume name (or number) of the volume you wish to have consolidated. The next question the system asks is something like the following:

```
From end of disk (block 27Ø)? (Y/N)
```

This question asks if you want the consolidation of the diskette to result in the free space being assigned to the blocks following block 27Ø, which, in this case, is at the end of the diskette. Normally, you would reply with a Y to this question, although, if your use of the Bad Blocks command shows some bad blocks, you might want to consign the free space in that area, then transfer the files on the diskette onto another diskette and Krunch the diskette again.

List Directory

You receive a list of the files on a volume when you use the List Directory command.

The List Directory command works exactly like the Extended Directory command, although it provides less information about the files and is therefore slightly faster than the Extended Directory command.

You should use the List Directory command if all you desire is a list of the files on a volume, while you use the Extended Directory command if you want to know about the size and location of a file on a volume.

Make

Some P-System programs require that a file of a fixed size already exist on the system, even if nothing is in that file. To create such a file, you use the Make command.

When you press M, for Make, you'll be asked what file name to make:

```
Make what file?
```

You should enter a valid P-System file name (remember, you can also enter the volume name and the file type, if desired). Immediately following the file name, you enter the number of blocks to create, enclosed in brackets. An example might look like this:

```
Make what file? OSBORNE:NEWFILE.TEXT[25]
```

In the above example, 25 blocks will be reserved for the file "NEWFILE.TEXT."

If you do not specify the number of blocks to reserve, the P-System looks for the largest set of contiguous blocks and assigns that to the file. Alternatively, you can assign half of the largest set of continguous space to a file by enclosing an asterisk in the brackets:

```
      Make what file? OSBORNE:NEWFILE.TEXT[*]
```

would assign 1Ø blocks to "NEWFILE.TEXT" if the largest free space on volume "OSBORNE" is 2Ø blocks.

If you accidentally Remove a file from the directory of the diskette, sometimes you can recover it by making a new file. The reason this works is that when you "make" a file, you really only create the directory entry; any information stored in the blocks for that file remains unchanged. The process to use to recover a file is to use the Extended Directory and Examine commands to find where the file you removed might be, then create a file using the Make command in a fashion that insures that the blocks of the old file that you found are assigned to the new file.

New

The New command clears the workfile. In other words, if there was a workfile in progress, it will be removed after you verify your action to the P-System:

```
      Throw away current workfile? (Y/N) Y
```

If you used the Get command to specify a workfile, theNew command cancels that previous command and resets the SYSTEM.WRK.TEXT as containing the current workfile.

Prefix

We've mentioned before that you can use the asterisk to substitute for the current, default volume name. You may have wondered if you can change that default.

The answer is that you can't. The asterisk refers to something called the "root volume," or the volume from which you started the system. But you can use another shorthand method of referring to volumes--the "prefix volume"--by typing only a colon. When you type a colon instead of a complete volume specification, the default prefix volume is used. You change the prefix volume with the Prefix command of the filer. Pressing P, for Prefix, results in the following message:

```
        Prefix is OSBORNE:

        Prefix titles by?
```

The above message tells you that "OSBORNE:" had been the prefix assumed by the system. You can now change that prefix to any valid P-System volume name.

We suggest that, after changing the date, you change the prefix volume to "#5:" (i.e., the second disk drive) before starting to use the system each day. Then, if you want to access something in drive A, you type *: before the file name, while if you want to access something in drive B, you simply type : and a file name.

Quit

The Quit option of the filer is easy to explain: it takes you out of the filer and returns you to the P-System command level. Need we say more?

Remove

To erase a file from the directory of a volume, you use the filer's Remove command.

When you press R, for Remove, the system answers with:

```
        Remove what file?
```

You may type any valid P-System file specification, including those using wildcard characters. The system locates the file (or files) you specified, then asks:

```
        Update directory?
```

Answering with a Y results in the deletion of the file (or files) in question; a N cancels the removal request.

Save

When you've completed use of a workfile, you should rename it with a name other than "SYSTEM.WRK.TEXT" or "SYSTEM.WRK.CODE." You do this using the filer's Save command.

When you press S, for Save, the system responds with:

```
        Save as what file?
```

The system is asking for the name you want to save the workfile under, so enter any valid P-System file specification.

Once the save operation is complete, a new, empty workfile is started. In short, Save is similar to performing a Change and a New command in quick succession.

Transfer

A command you'll use often is the Transfer one. As the name indicates, files are transferred from one volume to another (or the same volume) using this command.

Pressing T, for Transfer, results in the following message:

```
Transfer what file?
```

After you've entered a valid P-System file name, the system prompts you with:

```
To where?
```

Since there are many possibilities available to you using the Transfer command, we need to summarize their use for you:

<u>Copying a single file</u>: specify the volume and file name of the file to be moved in response to the first prompt, then the volume and file name for the destination. If the destination is to be the same as the same as the source (i.e., the same file name on two different diskettes), you may abbreviate the file name in the second response to just a dollar sign ($). Example:

```
Transfer what file?  *:THOMSFILE.TEXT

To where?  :$
```

The example puts the file "THOMSFILE.TEXT" on the second disk drive (:) with the same name, from the original on the first disk drive (*:).

<u>To copy a file on the same drive</u>: Respond as described above to the first prompt, then use the same volume name and file name in response to the second prompt, but include square brackets to indicate the number of blocks for the new file.

<u>To copy several files</u>: use the wildcard character in part of the file name you specify to the first prompt; use either the dollar sign ($) shorthand indicator or another wildcard-embedded file name in respnose to the second prompt.

To copy an entire diskette: first format a double density diskette using the CP/M COPY program, then use the Zero command to create the directory for the volume. Next, use the Transfer command, simply enter the volume names (no file names) for each of the prompts shown above. The system will ask you if you want all blocks transferred; you should verify your request by pressing a Y and RETURN. All files will be transferred from the first volume name you specify to the second. If, for some reason, the P-System finds a volume name on the destination diskette, you'll also be asked to verify that you want the volume destroyed.

Volumes

To find out what volumes currently exist on your system, you use the Volumes command. This one's an easy one, just press the V key, and the list of your volumes appear--that's all there is to it.

[NOTE: THERE IS CURRENTLY A BUG IN THE OSBORNE P-SYSTEM THAT MAKES IT SO THAT THE FIRST LINE OF THE VOLUMES LIST IS DESTROYED WHEN THE FILER PROMPTLINE REAPPEARS AT THE TOP OF THE SCREEN. TUT, TUT, TOM.]

The Volume command is handy for figuring out the name you assigned a diskette if you weren't clever enough to write it on the diskette label.

What

The What command tells you the current status of the workfile on the P-System. Four conditions are possible:

no workfile
workfile is FILENAME.TYPE
not named
not saved

As you may suspect, each of these conditions is unique:

No workfile means that no workfile has been specified using the Get command, and no SYSTEM.WRK.TEXT or SYSTEM.WRK.CODE file currently exists.

Workfile is FILENAME.TYPE tells you the name of the file the system considers the current Workfile.

Not named indicates that the file SYSTEM.WRK.TEXT or SYSTEM.WRK.CODE exists, but has not been named by a previous Get command.

Not saved indicates that the file SYSTEM.WRK.TEXT or SYSTEM.WRK.CODE exists, but has not yet been saved.

Examine

The Examine command in the filer is used to attempt to correct a problem with a file shown by the Bad Blocks command. In other words, if you use the Bad Blocks command and find that some errors exist on the diskette, you use the Examine command to attempt to correct the problem.

To execute the command, press X, for examine. The system displays the message:

```
Examine blocks on what vol?
```

As before, specify a valid P-System volume name, or use one of the shorthand methods of specifying a volume (*: or :). Next, the computer displays:

```
Block-range?
```

Here, you are to enter a range by typing the first block to examine, a hyphen, then the block number to end the examination with. Screen display erased before user can view it.

If the system finds a bad block, you will see a prompt that says:

```
Fix them?
```

Press Y to verify the request, N to cancel the operation.

Examine cannot always fix a bad block. The message

```
Mark as bad?
```

is displayed in this case. Be careful about hastily replying with a Y to this question. Once a block has been marked as bad by the P-System, it cannot ever be used again, and the information that is in that block is totally lost. Even the Krunch command, which rearranges files on the diskette, will have no effect on a marked bad block.

The best thing to do about bad blocks is to avoid them in the first place. Many P-System users execute the Examine command on any new diskette before they place any files on it. This assures that no physical defects on the diskette will show up later as bad blocks. This preventative measure, along with careful diskette handling, should make it so that you never encounter a bad block that can't be fixed.

Zero

The last filer command, Zero, is the one that creates the initial directory on a new diskette. After you press Z, for Zero, the computer will ask you several questions:

Zero dir of what vol?

Type the name of the volume you wish to have "zeroed," or use one of the shorthand characters (*: or :).

Destroy VOLUMENAME:?

You're asked to verify the action because you can use the Zero command to reinitialize a diskette that has been in use. The "VOLUMENAME:" indicates what volume name the P-System thinks the volume you specified currently has.

Are there xxx blocks on the disk? (Y/N)
or
# of blocks on the disk? whoops, no ^PB

If the diskette already had been used by the P-System, it will report how many blocks it thinks are on the diskette. You might not want to use a diskette that has many bad blocks, and therefore less storage capacity than normal.

If the diskette is new and hasn't been used previously, you need to type in the number of blocks to use. For Osborne 1 double density, that number is normally ###, but you could specify less. Whatever number you type, this is an irreversible decision once files are stored on the diskette (you must use the Zero command to re-initialize the diskette, thus destroying the directory).

msgs screwed up

New volume name?

Here's where you get to specify the name of the volume. We named our P-System diskette OSBORNE: so that you would know that it was the original from Osborne. We suggest you invent a naming pattern that is both consistent from diskette to diskette, and meaningful to you. Remember, volume names end with a colon.

blank screen, have to type return

NEWVOLUMENAME: correct?

Just to make sure that you didn't mistype the volume name, the P-System asks you to verify the name you just input. As normally, press Y to verify the action, N to cancel it.

You've now exhausted all of the filer commands. It's time to examine the other commands that are available to you.

The following command-level functions are of less~~or~~e interest to the P-System user than the filer, so we've given them a briefer treatment. Many users will find that they only need to know about the filer and the Execute functions.

Initialize and User Restart

The Initialize function has only one real purpose: to restart the system, or to place it in its initial state, as if you had just rebooted the system.

In actuality, Initialize causes the the system to look for a file called SYSTEM.STARTUP to be executed. Some programmers use this knowledge to create their own file of that name, in which will be instructions they want the computer to perform (such as display of a message). Initialize is executed any time any non-catastrophic (computer terminology ususally refers to these as "non-fatal" errors) error occurs. When the Initialize function is performed, most of the internal functions, variables, defaults, and so on, are set back to their original state.

For most users, therefore, pressing I at the command level, for initialize, is simply a quick way to restart the P-System or to reset the system to its initial state.

User Restart--pressing U at the command level--tells the P-System to rerun the last program executed. If you had previously run the filer, using User Restart would cause the filer to be rerun.

Halt

The Halt function does exactly what its name implies--it stops the system from doing any further execution of instructions.

Pressing H at the command level, for Halt, is a drastic action, since the only way you'll be able to get the computer's attention again is by pressing the Reset button on the front panel, then going through the entire startup process again (i.e., using PBOOT to start a P-System diskette).

Monitor

You can cause the P-System to remember all the commands you give it, and to save those commands in a .TEXT-type file you name.

The promptline for the Monitor looks like this:

```
Monitor: B(egin, E(nd, A(bort, S(uspend, R(esume
```

When you press B, for Begin, the system asks you for the name of a file to save the "monitoring" in. Once you've specified a name, all further input by you is saved in that file until you later invoke the Monitor and press either the E, A, or S keys. Each of the options other than Begin have obvious meanings:

E(nd	Completes the monitoring process and closes the file that was saving your input. You must End a monitoring session in order to have a usable input file.
A(bort	Cancels the monitoring process without saving anything into the Monitor file.
S(uspend	Temporarily suspends the monitoring process, but does not close the file you specified with the Begin command.
R(esume	Restarts the monitoring process after it has been suspended.

Monitoring commands do not appear or get saved in the file of input that is created.

At first, the concept of monitoring your input might not seem useful. It was added to the P-System for two basic purposes:

1. To allow the user to easily recreate an oft-repeated sequence of keystrokes. Instead of actually having to type it. To do so, when a filename is specified for execution, you add the option

   ```
   PI=filename
   ```

   where "filename" is the name of the file you saved the monitored session in.

2. To serve as a backup of the commands entered, just in case the results of a session need tobeduplicated(asinthecasewhenyou accidentally erase a newly created file).

Again, using the PI= option before execution of a program allows you to have the computer do the typing for you.

Debug

The Debug function available from the Command: promptline is similar to the DDT program CP/M uses; both allow you to examine and modify memory, as well as perform program execution in an extremely controlled manner.

Debug is a functional utility for programmers using the Development System, but most users of the P-System will never utilize Debug. We'll only briefly mention the commands that are made available. (Note: no promptline appears when Debug is active, so you'll need to consult this list until you learn the commands that are available.)

Q(uit		Quits the Debug module and returns to Command promptline.
R(esume		Exits Debug, although it remains active.
S(tep		Single steps through P-Code, then returns control to Debug.
B(reak		Specify segment, procedure and offset to stop execution at.
	S(et	Allows one of four breakpoints to be set.
	R(emove	Removes one of the four breakpoints.
	L(ist	Shows you a list of your breakpoints.
V(ariable		Specify variables or memory segments.
	G(lobal	Displays global memory.
	L(ocal	Displays local memory.
	I(nter	Displays intermediate memory.
	P(roc	Displays data segment of procedure.
	E(xtended	Displays variables of other segments.
C(hain		Changes reference for Variable command.
	U(p	Chains up mark stack links
	D(own	Chains down mark stack links
	L(ist	Lists current mark stacks.
E(nable		Enables display of List during single step.
D(isable		Disables display of List during single step.
L(ist		Lists specific data.
	R(egister	The registers are listed.
	P(code	The P-Code mnemonic is listed.
	M(arkstack	Mark stack is displayed

A(ddress	A specific address is listed.
E(very	All of the above display.

A(ddress	Displays a given address.
P(Code	Disassembles a given procedure.
M(emory	Allows following memory functions.

L(ock	Memory locks the Debug module.
S(wap	Swaps the Debug module

As you can probably tell from the listing, you need to have specific knowledge of the internals of the P-System to understand how to use Debug.

Other Functions and Error Messages

You'll note that we haven't described any other command-level functions, like Compile, Run, Execute, and Assembler. The reason we haven't is that they are not present on the P-System diskette you received. If you attempt to use them, you'll receive a message like:

```
SYSTEM.ASSEMBLER not found
```

Run gives same semantic as Compile

What is SYSTEM. PASCAL

C. Using the P-System

You're now ready to learn how you would use the P-System in day-to-day activity to run UCSD Pascal programs compiled on other systems.

C.1 How You Use It

So, you want to use the P-System, eh? Well, let's make sure that you have a reason to.

You don't normally need the P-System unless you have software that requires it to be present. Most users of the Osborne 1 computer probably use only CP/M-based programs. If all the software you've acquired uses CP/M, you can put your P-System diskette away and only come back to it and this manual when you purchase some software that uses the P-System.

A question comes to mind: If most users of the Osborne 1 simply utilize CP/M software, why does Osborne provide the P-System to all double density purchasers? The answer is simple: while the established software base for the Osborne is predominantly CP/M-based, other software, particularly that for the Apple and IBM personal computers, often uses the P-System--we wanted to make those programs available to you. Apple Pascal, for instance, is an earlier version of UCSD ~~Pascal,~~ and programs written for Apple Pascal are easily adoptable to the Osborne. You'll see more and more programs ~~in the fu~~ture using the P-System instead of CP/M.

adaptable?

Okay, you've received a piece of software that uses the UCSD P-System, how do you proceed?

First, you must determine the following: is the diskette format the software was provided to you the

> Osborne # double density format,
> UCSD P-System universal diskette format, or
> any other format?

If your answer is Osborne double density, then this subsection of the manual is relevant to you and you should follow the instructions here. If you answered UCSD P-System universal diskette format, proceed to the next subsection and follow the instructions there. If the diskette you have is in any other format, you'll have to get it converted by a friend, your dealer, or your local user group, to one of the Osborne-usable formats. (Note: if the program was supplied to you on paper in source-code form, you might want to purchase the UCSD Development System from the Osborne Approved Software program; use the editor to enter the program, and then the compiler to make it usable.)

Since you're still reading, we'll assume that you have a diskette in Osborne double-density format with UCSD P-System programs on it. Your first step should be to make a copy of the diskette using the Osborne double density COPY program. NEVER use the original diskette you receive for any purpose other than to make a working copy.

Once you have a copy, here's the sequence of events you'll want to use to create a working diskette:

Seems messy having to shuffle 3 diskettes into Dr A

1. Boot double density CP/M. — vanilla CP/M ~~of the one in which p-system comes?~~

2. Use the program PBOOT.COM (supplied on the P-System diskette you received with double density) to start the UCSD P-System. You do this by typing Switch disks?

   ```
   PBOOT
   ```

   followed by a RETURN when you see the A> prompt.

3. Put a copy of the P-System diskette you created in the first section of this manual in drive A, and your newly acquired P-System program diskette in drive B. Press RETURN, as instructed by the message on the screen.

4. After a few ~~moments~~ minutes! the UCSD P-System sign-on message appears, as shown below:

   [sign-on]

5. To see what's on the diskette in drive B, do the following:

   ```
   Press F, for filer
   Press L, for list directory
   Type #5: followed by a RETURN
   ```

   A list of files on the diskette should appear. ~~No directory appears~~

6. Normally, you'd want to compact the files onto one diskette, if possible, so the next steps are to perform the transfer to your P-System diskette, as follows:

   ```
   Press F, for filer (if you're not already there)
   Press T, for transfer
   Type ?,#5:? followed by a RETURN
   ```
   whoops, ^PB missing

   This last command tells the P-System to transfer all files on the diskette in drive B (device #5:) to the current device (the diskette in drive A). For each file you will be asked to verify the request. You do so by typing Y.

Your new files we be transferred to your P-System diskette.

All Osborne-supplied P-System software comes with a description of the process you should go through to make copies and prepare usable working diskettes.

To function, you must have the following P-System files on the diskette you use PBOOT on:

1 SYSTEM.STARTUP
2
3
4

Normally, you would also have your program files on this diskette, which goes into drive A. You would prepare a blank, formatted diskette to be used for saving data files to place into drive B.

To use your program, you'd first press X, for execute, then enter the name of the program when prompted to do so by the system. Sometimes, however, the programmer might include a special SYSTEM.STARTUP file that will make it so that the program is run automatically when you first use PBOOT to start the P-System (very much like AUTOST.COM loads and executes a program in CP/M).

Due to the 184K disk capacity of the Osborne drives, you may find that some programs intended for machines that come with larger capacity drives will not allow all the program files needed to fit onto the diskette in drive A. In such a case, you would put the remaining files on the data diskette in drive B, but this obviously reduces the amount of data that can be stored on that diskette.

To recap, in normal operation, you use the Osborne-supplied diskettes to create a P-System diskette onto which you also copy the P-System programs you receive. You also format a blank diskette and place it in drive B. Most programs you will encounter function in this manner.

C.2 The Universal Diskette Format

As described in your Double Density User's Guide, no standard exists for 5 1/4-inch diskettes. That's not entirely true, however, as that statement applies to the CP/M operating environment.

Osborne, IBM, Softech Microsystems (the originator of the P-System), and several other companies all use a common "Universal" diskette format for transferring programs between machines.

This format is basically the IBM single-density diskette format (approximately 156K of data storage capacity on a single diskette). While you could use programs on diskettes in this format within any Osborne 1 computer equipped with double density--the system automatically recognizes that such a diskette has been inserted--we recommend that you immediately follow the instructions in the section above to create working Osborne-format P-System diskettes. The rationale for this suggestion is simply that the Osborne format provides about 2ØK more diskette storage capacity (about 12 percent more) than the Universal standard format. Most programs that use the P-System make good use of the extra 4ØK (two diskettes) the Osborne format provides.

You should be aware of one limitation of the current Osborne P-System, it cannot be used to create new, formatted, Universal format diskettes. You can read from and write to Universal format diskettes you receive from others, but you cannot create new ones. This minor deficiency will be corrected with a future release of the P-System.

The Universal diskette format does allow you immediate access to a number of P-System programs, even those written on IBM and other computers that have different CPUs!

The idea of a Universal diskette format may not sound like a overwhelming innovation, but for the small computer marketplace, it is. The more companies that support the Universal format, the more software you'll find available for your Osborne. Therefore, if you encounter software that you'd be interested in, but isn't in the Universal format, tell the software creator that you'd prefer the product in the standard format.

C.3 An Example

The following sequential screens show an example of a session in which a user creates working P-System diskettes from a diskette received from another software company. We've added a few annotations to the printouts so that you can follow what's happening.

If you still don't understand the process, you should carefully compare the following example with the description of what you should do that appears in subsection C.1.